Number Theory

The study of integers

Important applications

- Division and modular arithmetic
  - Generating pseudorandom numbers
  - Check digits
  - Encryption
  - Hashing
    - Assigning memory locations
  - Arithmetic with LARGE integers. (Too large to be represented easily in a computer)
- Prime numbers
  - Modern cryptography

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Division

If a and b are integers and $a \neq 0$, we say that a **divides** b if there is an integer c such that $\boldsymbol{b = ac}$.

In other words, $\frac{b}{a}$ is an integer. There is no remainder.

$a \mid b$ means a divides b

$3 \mid 12$ is true because $\frac{12}{3} = 4$ with no remainder.

$3 \mid 13$ is not true because $\frac{13}{3} = 4$ remainder 1 or $4.33\overline{3}$.

Linear Combinations

If a number x divides two different numbers, y and z, then x divides **any** linear combination of y and z.

A **linear combination** of two numbers is the sum of multiples of those numbers. For example, $3x + 2y$ and $-7x + 18y$ are both linear combinations of x and y.

Let x, y, and z be integers. If $x|y$ and $x|z$, then $x|(sy + tz)$ for **any** integers s and t.

Using quantifiers:

Let x, y, z, s, and t be integers. $\forall s \forall t \left((x|y \land x|z) \rightarrow (x\,|(sy + tz)\right)$

Explore in Python

Division Theorem

Dividing an integer by a positive integer produces a quotient q and remainder r.

Let n and d be integers. Then there exist unique integers q and r where $0 \leq r < d$ such that $\boldsymbol{n} = \boldsymbol{dq} + r$.

This is a fancy way of saying that if you divide one integer by another integer, there will be an integer quotient q and a remainder r. (r might be 0)

$$q = n \text{ div } d = \left\lfloor \frac{n}{d} \right\rfloor \qquad r = n \bmod d = n - d\left\lfloor \frac{n}{d} \right\rfloor$$

Python:

```
q = n // d          r = n % d
```

Example:
$n = 101, d = 11$

$q = 101 \text{ div } 11 = \left\lfloor \frac{101}{11} \right\rfloor = 9$
$r = 101 \bmod 11 = 101 - 11 * 9 = 2$

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Compute q and r algorithmically

Case 1: n ≥ 0.	Case 2: n < 0.

q := 0
r := n
While (r ≥ d)
    q := q + 1
    r := r - d
End-While
	q := 0
	r := n
	While (r < 0)
	    q := q - 1
	    r := r + d
	End-While

Use these steps to compute q and r for the following:

$n = 29, d = 7$
$q = 4, r = 1$

$n = 58, d = 9$
$q = 6, r = 4$

$n = -29, d = 7$
$q = -5, r = 6$

$n = -58, d = 9$
$q = -7, r = 5$

Irani, Sandy. *Discrete Mathematics*, Zyante Inc. (zyBooks.com)

Modular arithmetic

- Addition
- Multiplication

When performing modular arithmetic, the result is always performed under $mod\ m$ for some number m.

This means the result is always less than m and greater than or equal to 0.

Example:
$(6 + 5) \bmod 7 = 11 \bmod 7 = 4$

Example:
$(6 * 5) \bmod 7 = 30 \bmod 7 = 2$

Congruences

$a \equiv_m b$ — a is congruent modulo m to b

$a \equiv b \pmod{m}$ — a is congruent modulo m to b

Both a and b have the same remainder when divided by m.

Example:

14 and **21** are congruent modulo **7** because 14 mod 7 = 21 mod 7 = 0

0, 5, 10, 15, 20 are all congruent modulo **5**.

So are **1, 6, 11, 16, and 21**.

Congruences

Another way to think about it:

$$a \bmod m = b \bmod m$$

0, 12, 24, 36, 48, … are all congruent mod 12

0, 7, 14, 21, 28, … are all congruent mod 7

1, 8, 15, 22, 29… are all congruent mod 7

Equivalence Classes

$[a]_R = \{s \mid (a, s) \in R\}$

Meaning: If we have an equivalence relation R on set A, the set of all elements s related to an element a of A is the *equivalence class*.

In other words, if we have (a, b) as a pair in R, then the set of all b for a given a is the equivalence class of that a.

Example:

Let R be the equivalence relation **congruence modulo 4** where the domain is $\mathbb{Z}$.

What is $[0]_R$?

$[0]_4 = \{\dots, -8, -4, 0, 4, 8, 12, \dots\}$

What is $[1]_R$?

$[1]_4 = \{\dots, -7, -3, 1, 5, 9, \dots\}$

Example:

Let R be the equivalence relation on A for **congruence modulo 5** where A is the set of integers defined by `range(-20,21)`.

Use Python to generate the equivalence class $[3]_5$

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Computing arithmetic operations mod m.

Let m be an integer larger than 1. Let x and y be any integers. Then,

$$[(x \; mod \; m) + (y \; mod \; m)] \; mod \; m = [x + y] \; mod \; m$$

$$[(x \; mod \; m)(y \; mod \; m)] \; mod \; m = [xy] \; mod \; m$$

Let's do Challenge 9.2.1

Computing exponents:

$46^{10} \bmod 7$

$= (46 \bmod 7)^{10} \bmod 7$

$= 4^{10} \bmod 7$

$4^1 \bmod 7 = 4$

$4^2 \bmod 7 = 4^1 * 4^1 \bmod 7 = 4 * 4 \bmod 7 = 16 \bmod 7 = 2$

$4^4 \bmod 7 = (4^2 \bmod 7) * (4^2 \bmod 7) = 2 * 2 \bmod 7 = 4 \bmod 7 = 4$

$4^8 \bmod 7 = (4^4 \bmod 7) * (4^4 \bmod 7) = 4 * 4 \bmod 7 = 16 \bmod 7 = 2$

$4^{10} \bmod 7 = (4^8 \bmod 7) * (4^2 \bmod 7) = 2 * 2 \bmod 7 = 4 \bmod 7 = 4$

Since $4^{10} \bmod 7 = 4$, then $46^{10} \bmod 7 = 4$

Pseudorandom Number Generators

One of the common ways to generate pseudo-random numbers is by using a **linear congruential generator** (LCG).

This generator is a recurrence relation defined as:

$$X_{n+1} = (aX_n + c) \bmod m$$

The values for a, c, and m must be chosen carefully.

For more reading, see Linear congruential generator on Wikipedia.

See a LCG implemented in Python